Spot it!

Analyzing my 2024 Spotify listening data

Jenn Leishman

2025-05-05

Contents

Introduction

I think it's a naturally human thing to be curious about your own habits and data. As people, everything we do everyday contributes to some sort of data that, though not collected, can be transferred to physical statistics. Some of these are collected using modern technology, including how many steps you take, or your heartrate throughout the day. Others are self-recorded, and many people have taken to apps such as Goodreads or Letterboxd to record and rate their media consumption.

For music listeners, Spotify, a popular music streaming platform, can collect statistics for you. The app produces an annual "Spotify Wrapped", which produces a slideshow of personalized statistics and top songs.

This report is inspired by that concept.

I was curious to download and analyze my own Spotify data, and see what patterns I could observe using the Tidyverse and data wrangling/modelling skill I have gained. I learned that I could download my own Spotify data in a JSON format, which I learned could actually be read into R using the jsonlite package.[1]

My data

As I mentioned, my data file was downloaded from Spotify in a JSON format. I read in my data into a dataframe, and then used the head and glimpse functions to view it.

```
json <- fromJSON("StreamingHistory_music_01.json")
song_data <- as.data.frame(json)
head(song_data)
```

```
##            endTime      artistName                          trackName msPlayed
## 1 2024-01-01 23:42   Whatever, Dad           Death of the Phone Call     2035
## 2 2024-01-03 20:09   Whatever, Dad           Death of the Phone Call     1344
## 3 2024-01-03 22:51   Whatever, Dad           Death of the Phone Call    91652
## 4 2024-01-03 22:56 Phoebe Bridgers                      Scott Street   305000
## 5 2024-01-03 22:59   Julia Jacklin to Perth, before the border closes   117133
## 6 2024-01-04 01:52   Julia Jacklin to Perth, before the border closes    59866
```

```
glimpse(song_data)
```

```
## Rows: 8,924
## Columns: 4
## $ endTime    <chr> "2024-01-01 23:42", "2024-01-03 20:09", "2024-01-03 22:51",~
## $ artistName <chr> "Whatever, Dad", "Whatever, Dad", "Whatever, Dad", "Phoebe ~
## $ trackName  <chr> "Death of the Phone Call", "Death of the Phone Call", "Deat~
## $ msPlayed   <int> 2035, 1344, 91652, 305000, 117133, 59866, 192070, 220244, 8~
```

As you can see, the data has 4 columns and 8,924 rows. These columns are endTime, containing the date and time I finished the song (in character format), artistName, trackName, and msPlayed (milliseconds played). I honestly was hoping that my data would give me more song information, such as album, genre or song length, however this was not the case.

Given the limited information included in my data, I actually attempted to access the Spotify API and use it in R. Unfortunately, though this is hypothetically possible, as the Spotify API would contain further specific song information, I was not able to get working code and access to the API in R, as it takes certain developer accounts and access codes. I also could not find a comprehensive only song database to use that not too outdated to join with my Spotify data.

This leaves us with my data, stored in a dataframe titled song_data. Though I'm limited in my ability to identify different genre patterns in my listening, I still am able to use the dates to my advantage. Also, each

[1] https://datacarpentry.github.io/r-socialsci/07-json#:~:text=You%20can%20read%20the%20JSON,does%20not%20download%20the%20file.

time I listened to a song is a different row entry in the dataframe, which means I do have a means of figuring out how many times I listened to a song.

Guiding research topic

Does time of day or time of year have an impact on the amount of music I listen to?

I really like the focus of my topic being on predicting my own habits. Data wrangling and modelling can be very useful in all areas, however, can also be applied to understand patterns of yourself for personal use and analysis.

Wrangling and visualization 1: Top artists

The first element of data wrangling I set myself up with was mostly for fun rather than to aid in my research question.

By using the Tidyverse to find my most listened to artists of 2024, I was able to learn new graphing techniques and also practice some basic data wrangling skills.

First, even though I know it's unconventional in "tidy" data, I decided to create a key in order to identify unique songs. If I just used the trackName variable, there is a high chance that there are multiple songs I listened to throughout the year with the same name. Therefore, I created a new column titled "fullKey" and in that, used a stringr function to combine the trackName and artistName.

```
# Create a key to identify song title/artist pairs
song_data_w_key <- song_data|>
  mutate(fullKey = str_c(trackName,",",artistName))
```

Next, I needed to define what it means to "listen" to a song or an artist. For the sake of this specific analysis, I decided to only count songs I listened to more than 50% of.

How, Jenn, would you be able to do this if the full length of the song is not included in the dataframe?

Great question! I don't actually know the full length of the song. That being said, I can make an edcuated guess using the msPlayed variable. A majority of songs I lisetned to, I would have listened to them all the way through at least one time in the past year. Of course, there are some songs I just never listened to all the way through, but logically, these would be obscure and not end up in my top songs or artists anyways.

Using this logic, I decided to assume that the total song length was equivalent to the maximum msPlayed for the song. I then could filter it by song instances where the msPlayed was greater than or equal to 50% of the total song length.

```
# Making sure I only count songs where I listened to 50% or more
song_data_time <- song_data_w_key |>
  group_by(fullKey) |>
  mutate(totalSongLength = max(msPlayed)) |>
  filter(msPlayed >= 0.5*totalSongLength)
```

I could then count the number of times I listened to artists based on the number of times they appeared in my cleaned dataframe, and create a count column. I sliced the dataframe to only get the top 10 artists with the highest counts, who would be the 10 artists I listened to the most in 2024.

Furthermore, I wanted to use images in my visualization of this cleaned data. Therefore, I created a new column titled "image" to store the image files saved for each artist.

```
top_artists <- song_data_time |>
  group_by(artistName) |>
  mutate(count = n()) |>
  arrange(desc(count)) |>
  ungroup() |>
  distinct(artistName, .keep_all = TRUE) |>
  slice_head(n = 10) |>
  mutate(
    image = c("adriannelenker.jpeg","phoebebridgers.jpeg","adammelchor.jpg",
              "chappellroan.jpeg","taylorswift.webp","boygenius.jpg",
              "lucydacus.jpeg","tophouse.webp","fleetwoodmac.jpg",
              "hozier.jpg")
  )
```

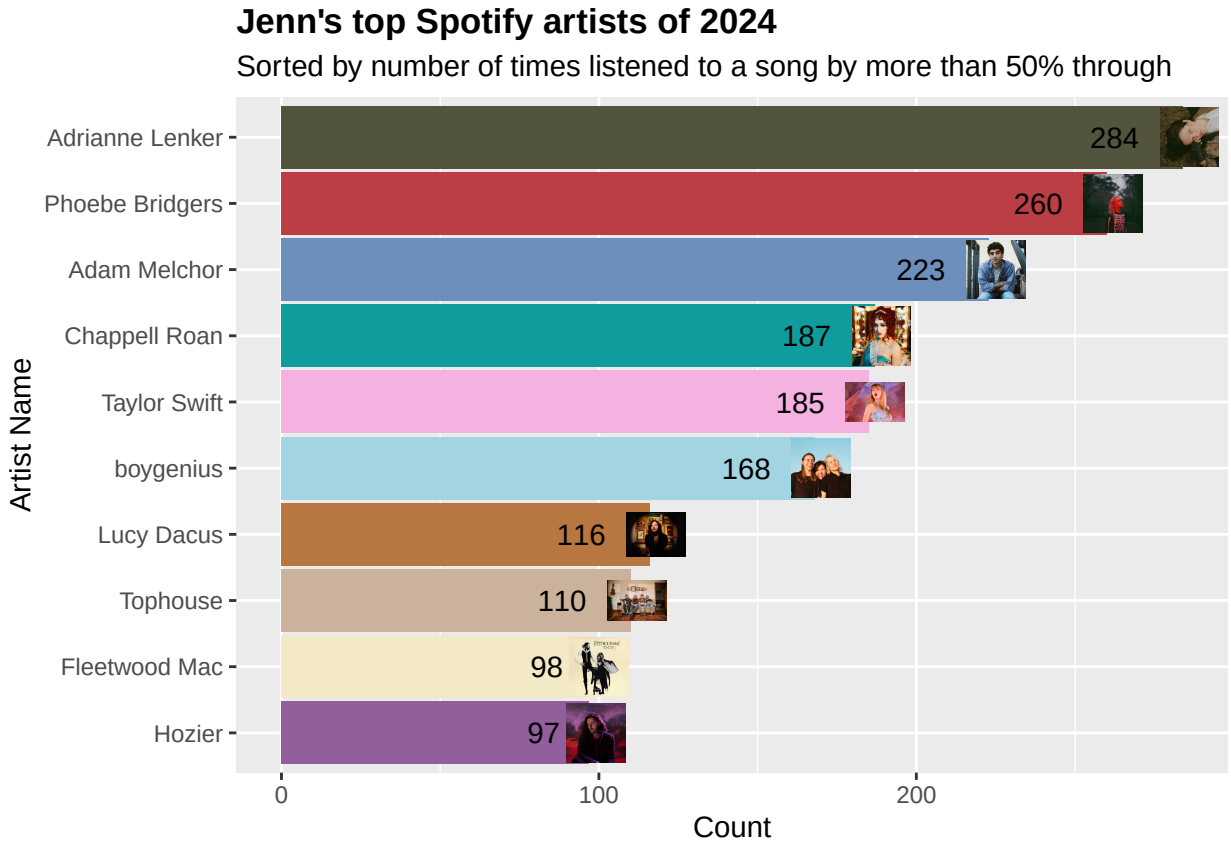
```
top_artists
```

```
## # A tibble: 10 x 8
##    endTime     artistName trackName msPlayed fullKey totalSongLength count image
##    <chr>       <chr>      <chr>        <int> <chr>             <int> <int> <chr>
##  1 2024-01-11~ Adrianne ~ Indiana     357138 Indian~          357138   284 adri~
##  2 2024-01-04~ Phoebe Br~ Savior C~   157757 Savior~          241640   260 phoe~
##  3 2024-01-12~ Adam Melc~ Why You?    160613 Why Yo~          265082   223 adam~
##  4 2024-01-22~ Chappell ~ Red Wine~   192720 Red Wi~          301033   187 chap~
##  5 2024-01-18~ Taylor Sw~ Nothing ~   258813 Nothin~          343021   185 tayl~
##  6 2024-01-04~ boygenius  Bite The~   192070 Bite T~          381005   168 boyg~
##  7 2024-01-04~ Lucy Dacus I Don't ~   163292 I Don'~          325312   116 lucy~
##  8 2024-07-08~ Tophouse   Number O~   196000 Number~          196000   110 toph~
##  9 2024-01-17~ Fleetwood~ Silver S~   288786 Silver~          288786    98 flee~
## 10 2024-01-05~ Hozier     Would Th~   268293 Would ~          268293    97 hozi~
```

Now we could plot! In order to include the images, I used the geom_image ggplot function.[2]

```
ggplot(top_artists, aes(x = reorder(artistName, count), y = count, fill = artistName)) +
  geom_bar(stat = "identity", width = 0.95) +
  geom_image(aes(image = image), size = 0.087, nudge_y = 2) +
  coord_flip() +
  labs(
    title = "Jenn's top Spotify artists of 2024",
    subtitle = "Sorted by number of times listened to a song by more than 50% through",
    x = "Artist Name",
    y = "Count"
  ) +
  theme(plot.title = element_text(face = "bold")) +
  scale_fill_manual(values = c(
    "#6c8fbb",
    "#52543e",
    "#a3d4e2",
    "#0f9c9c",
    "#f4e9c7",
    "#925e9b",
    "#b87640",
    "#ba3e44",
    "#f5b1e0",
    "#cab29d"
  )
  ) +
  guides(fill = FALSE) +
  geom_text(aes(label = count), position = position_dodge(0.9), hjust = 1.9)
```

[2]https://www.youtube.com/watch?v=mKOQCMlNnt0



As I mentioned, this was mostly just a fun experiment for myself. However, this graph does reveal the count of different artists, count being how many times I listened to a song of theirs, and reveals my top artists. If you don't know their music, it likely doesnt' mean a lot to you, but serves as an intro into my data.

Wrangling and visualization 2: Seasons

For my next visualization, I decided to start from scratch with my song_data dataframe, and re-wrangle it for a different purpose. More related to my research question, I decided that I wanted to visualize how season impacted the amount of music I listened to.

I used my data as well as an R package called lubridate[3] in order to pull the month from the timestamp.

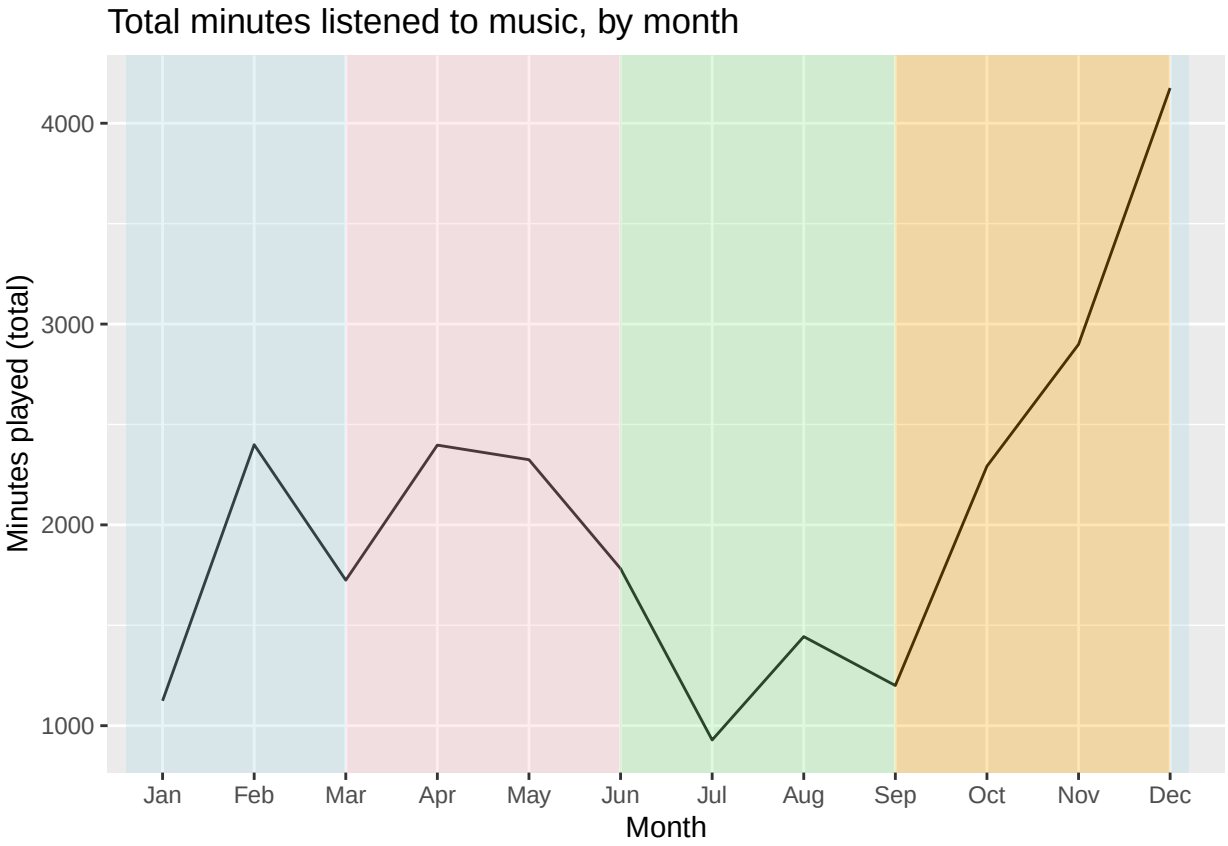
I also used mutate to calculate how many minutes of music I played each month, and then converted that from miliseconds to minutes for easier user comprehension.

```
songs_seasons <- song_data |>
  mutate(monthAbb = month(endTime, label = TRUE)) |>
  mutate(month = month(endTime)) |>
  group_by(month) |>
  mutate(msPlayedByMonth = sum(msPlayed)) |>
  mutate(minPlayedByMonth = msPlayedByMonth / 60000) |>
  select(monthAbb, minPlayedByMonth)
```

For my graph itself, I learned how to use a ggplot function annotate, because I wanted to color code the graph based on season to see spikes in season.[4] This way, we can see how the minutes listened changes from winter to spring to summer to fall.

```
songs_seasons |>
  ggplot(aes(x = monthAbb, y = minPlayedByMonth)) +
  geom_line(group = 1) +
  annotate("rect", xmin = 0.6, xmax = 3, ymin = -Inf, ymax = Inf, fill = "lightblue",
   ↪  alpha = 0.3) +
  annotate("rect", xmin = 3, xmax = 6, ymin = -Inf, ymax = Inf, fill = "pink", alpha =
   ↪  0.3) +
  annotate("rect", xmin = 6, xmax = 9, ymin = -Inf, ymax = Inf, fill = "lightgreen",
   ↪  alpha = 0.3) +
  annotate("rect", xmin = 9, xmax = 12, ymin = -Inf, ymax = Inf, fill = "orange", alpha =
   ↪  0.3) +
  annotate("rect", xmin = 12, xmax = 12.2, ymin = -Inf, ymax = Inf, fill = "lightblue",
   ↪  alpha = 0.3) +
  labs(
    title = "Total minutes listened to music, by month",
    x = "Month",
    y = "Minutes played (total)"
  )
```

[3] https://lubridate.tidyverse.org/reference/lubridate-package.html
[4] https://stackoverflow.com/questions/9178024/ggplot2-shade-area-between-two-vertical-lines



Looking at this graph, we can see that my music listening is not very consistnetly month-to-month, which does contribute to my research question curious about changes over the year. THe biggest spike in the graph is in December, though my music listening decreased very strongly in the summer and increased very strongly in the fall.

To continue following these patterns, I wanted to look more specifically at the entire year.

Wrangling and visualization 3: Day/hour heatmap

In order to look more specifically at the entirety of 2024 rather than just individual seasons and months, I wanted to also include hour of day as a contributing variable.

My first step was, again, to go back to my initial song_data and re-wrangle for how I wanted it. I will use lubridate again to get the months, and convert them to factors in order to order them correctly in my graph. I actually use the stringr function str_split_fixed in order to get the day of the month as well as the hour of day.^ [https://stringr.tidyverse.org/reference/str_split.html]

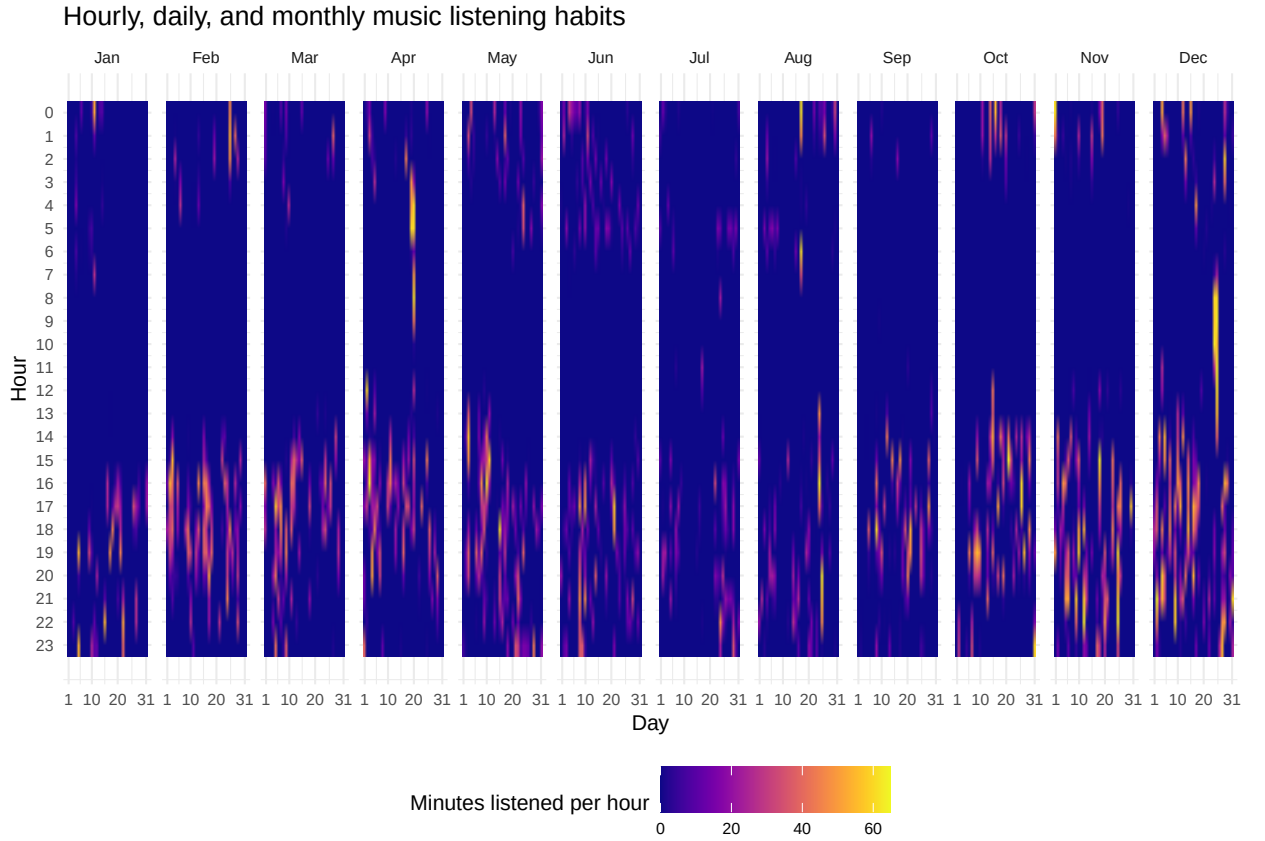
```
# Get date elements
songs_date <- song_data |>
  mutate(month = month(endTime, label = TRUE)) |>
  mutate(date = str_split_fixed(endTime, " ", 2)[, 1],
         day = as.integer(str_split_fixed(date, "-", 3)[, 3]),
         time = str_split_fixed(endTime, " ", 2)[, 2],
         hour = as.integer(str_split_fixed(time, ":", 2)[, 1])
         ) |>
  group_by(month, day, hour) |>
  summarize(minsPerHour = sum((msPlayed / 60000), na.rm = TRUE), .groups = "drop") |>
  mutate(month = factor(as.character(month), levels =
   ↪  c("Jan","Feb","Mar","Apr","May","Jun",
"Jul","Aug","Sep","Oct","Nov","Dec")))
```

For the type of graph, I decided that I wanted a very physical visualization of the times I listened to more music. I chose to create a heatmap, heavily relying on code from r-graph-gallery.com[5]

```
# heatmap

full_grid <- expand_grid(
  day = 1:31,
  hour = 0:23,
  month =
   ↪  factor(c("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"),
   ↪  levels =
   ↪  c("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec")))

songs_filled <- full_grid |>
  left_join(songs_date, by = c("day", "hour", "month")) |>
  mutate(
    minsPerHour = replace_na(minsPerHour, 0)
  )

ggplot(songs_filled, aes(day,hour, fill = minsPerHour)) +
  geom_raster(interpolate = TRUE) +
  scale_fill_viridis(name = "Minutes listened per hour", option = "C") +
  facet_grid(~month) +
  scale_y_continuous(trans = "reverse", breaks = unique(songs_date$hour)) +
  scale_x_continuous(breaks = c(1,10,20,31)) +
  theme_minimal(base_size = 8) +
  labs(
    title = "Hourly, daily, and monthly music listening habits",
    x = "Day",
    y = "Hour") +
```

[5]https://r-graph-gallery.com/283-the-hourly-heatmap.html

```
theme(legend.position = "bottom")
```



As you can see, the heatmap shows how many minutes of music I listened to each hour, each day, and each month during 2024. The darker blue means the less music I listened to, while the lighter pink and yellow spots, or the "warmer" colors, provide visualization for times when I listened to more music, the brightest yellow being 60 minutes, or the entire hour. You can definitely notifce that I listened to more music often in the afternoons and evenings, and any music I listened to in the early hours of the day seems to continue from the previous night. I also had a big spike of music listening earlier in the day during mid December, which I find interesting. You can see similar patterns from my seasons graph in that the colors for summer months seem to be a lot lighter, though following similar hourly patterns as the rest of the year. There's also a spike of music listening during April that this graph reveals, but at kind of odd and middle hours of the morning.

Logistic Regression Models

ROC curve with month + hour predictor

Because of the interesting patterns found in my heatmap, I decided I wanted to see how strong hour of day and month are as predictors for my music listening habits.

First, I took my songs_date dataframe that I used for my heatmap, containing individual days, months, and hours, and created a boolean column called minsAbove45. Essentially, I decided that any hour I listened to 45 minutes or more of music could be then considered as an hour during which I listened to a significant enough amount of music to be an hour listened to music. I'm aware this is a bit arbitrary, but I don't particularly care the exact number of minutes I listened to, rather, times of day where I do listen to music significantly.

Here, I also filtered NA values that were negatively impacting my results.

```
songs_classification <- songs_date |>
  mutate(minsAbove45 = case_when(
    minsPerHour <= 45 ~ 0,
    minsPerHour > 45 ~ 1
  ))

songs_classification <- na.omit(songs_classification)

songs_classification <- songs_classification |>
  filter(!is.na(minsAbove45))
```

```
# Splitting my data into training (80%) and testing (20%) sets

set.seed(3072) # random number chosen as seed
train_indexes <- as.vector(createDataPartition(songs_classification$minsAbove45, p = 0.8,
 ↪  list = FALSE))
music_train <- slice(songs_classification, train_indexes)
music_test <- slice(songs_classification, -train_indexes)

dim(songs_classification)
```

```
## [1] 1330    5
```

```
dim(music_train)
```

```
## [1] 1064    5
```

```
dim(music_test)
```

```
## [1] 266   5
```

```
fit <- glm(minsAbove45 ~ hour + month, data = music_train, family = "binomial")
```

```
# Creating a confusion matrix for my data

threshold <- 0.2

predicted_prob_train <- predict(fit, newdata = music_train, type = "response")
predicted_prob_test <- predict(fit, newdata = music_test, type = "response")

predicted_class_train <- factor(if_else(predicted_prob_train > threshold, "1", "0"),
 ↪  levels = c("0", "1"))
```

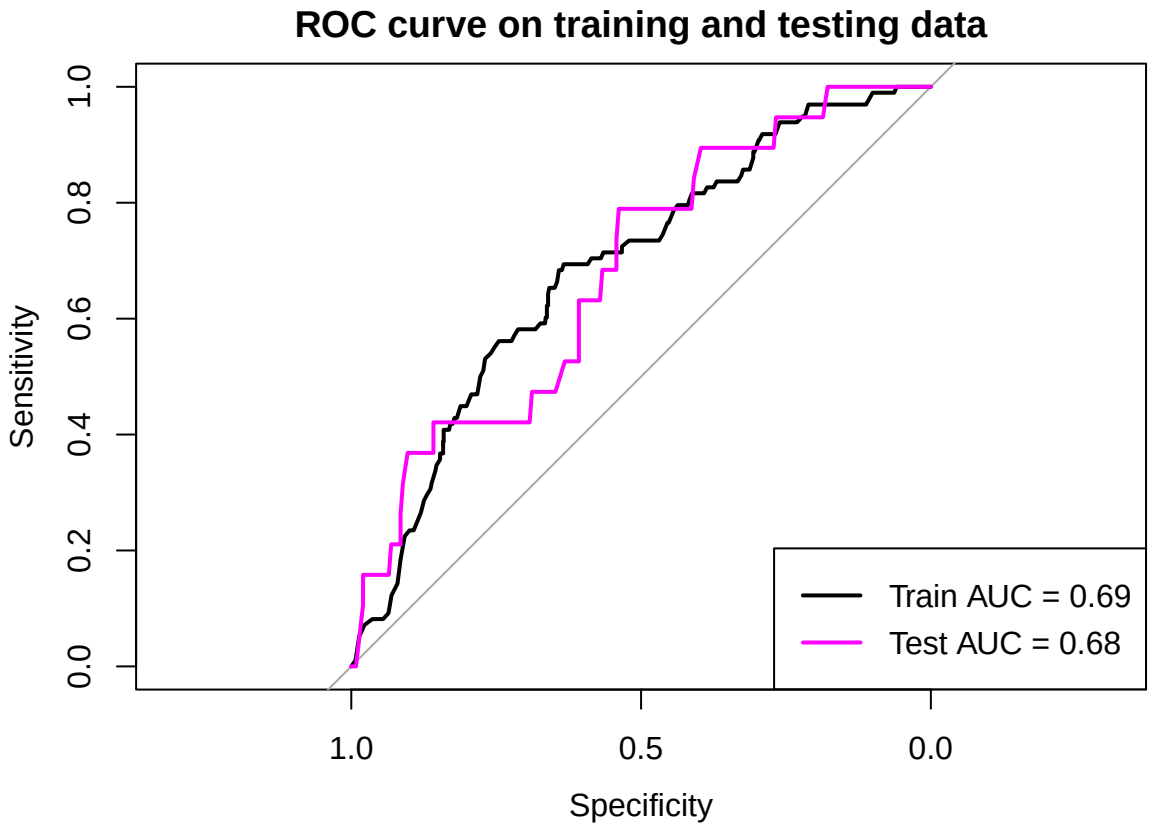
```
predicted_class_test <- factor(if_else(predicted_prob_test > threshold, "1", "0"), levels
 ↪  = c("0", "1"))

music_train$minsAbove45 <- factor(music_train$minsAbove45, levels = c("0", "1"))
music_test$minsAbove45 <- factor(music_test$minsAbove45, levels = c("0", "1"))

confusion_matrix_train <- confusionMatrix(
  data = predicted_class_train,
  reference = music_train$minsAbove45,
  positive = "1"
)

confusion_matrix_test <- confusionMatrix(
  data = predicted_class_test,
  reference = music_test$minsAbove45,
  positive = "1"
)

confusion_matrix_train$byClass["Sensitivity"]
```

```
## Sensitivity
##  0.07142857
```

```
confusion_matrix_test$byClass["Sensitivity"]
```

```
## Sensitivity
##   0.1052632
```

```
# Plotting the ROC curve
roc_train <- roc(music_train$minsAbove45, predicted_prob_train)
roc_test <- roc(music_test$minsAbove45, predicted_prob_test)

plot(roc_train, col = "black", lwd = 2, main = "ROC curve on training and testing data")
plot(roc_test, col = "magenta", lwd = 2, add = TRUE)

legend("bottomright",
 legend = c(
 paste("Train AUC =", round(auc(roc_train), 2)),
 paste("Test AUC =", round(auc(roc_test), 2))
),
 col = c("black", "magenta"),
 lwd = 2
)
```



We can see that I have a train AUC of 0.69, and a test AUC of 0.68 (very similar). The closer the AUC is to 1, the closer the model is to "perfect". 0.69 isn't necessarily amazing, however, it's significantly over 0.5 (the random point), and therefore, reveals that hour + month can somewhat resonably predict if I listened to music.

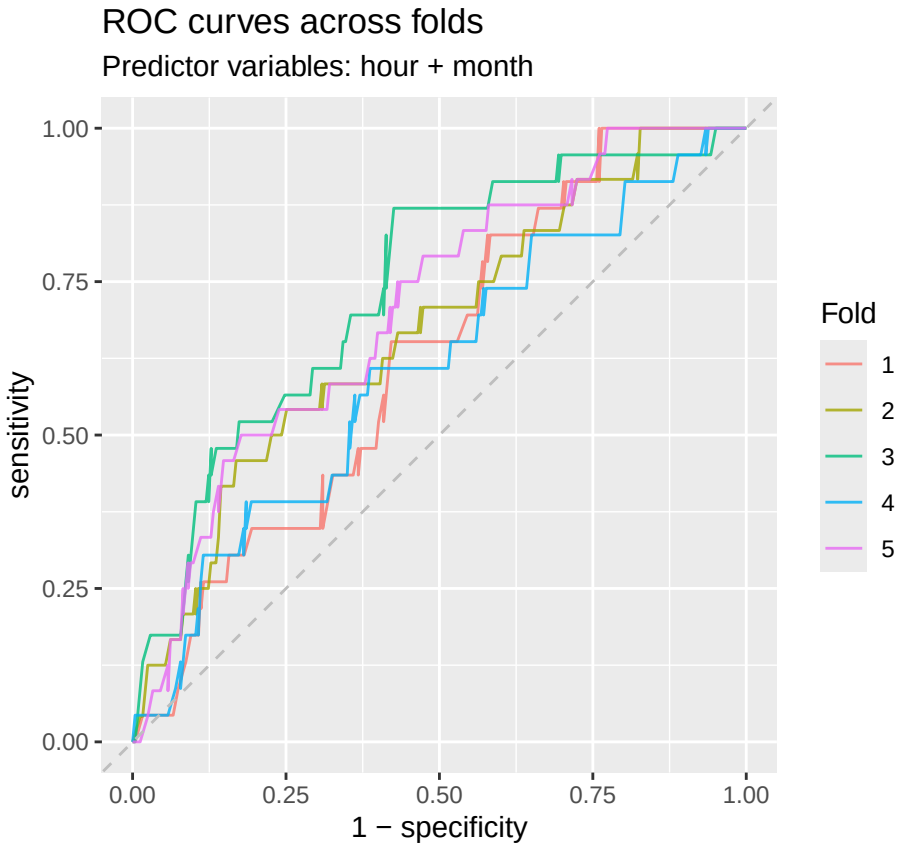
K-Fold cross validation for hour + month predictors

Next, I decided to peform a k-fold cross validation using 5 folds.

```
songs_classification$minsAbove45 <- factor(
  songs_classification$minsAbove45,
  levels = c(0, 1),
  labels = c("no", "yes")
)

cv_control <- trainControl(
  method = "cv",
  number = 5,
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = "final"
)

cv_model <- train(
  minsAbove45 ~ hour + month,
  data = songs_classification,
  method = "glm",
  family = "binomial",
  trControl = cv_control
)

roc_list <- cv_model$pred |>
  group_by(Resample) |>
  group_map(~ roc(response = .x$obs, predictor = .x$yes, levels = c("no","yes")))
```

```
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
```

```
roc_df <- map_dfr(roc_list, ~ as_tibble(coords(.x, "all")), .id = "Fold")

ggplot(roc_df, aes(x = 1 - specificity, y = sensitivity, color = Fold)) +
  geom_line(alpha=0.8) +
 geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "gray") +
 labs(
   title = "ROC curves across folds",
   subtitle = "Predictor variables: hour + month") +
 coord_equal()
```



As we can see, all the folds remain relatively consistent to our original ROC curve.

ROC curve with just hour predictor

Now, I want to test hour and month as predictors separately from one another, primarily to gauge the impact they have. I will be repeating the code for both the ROC curve and the k-fold cross validation in the exact same way with the same seed, just with the indvidual predictors.

```
songs_classification <- songs_date |>
  mutate(minsAbove45 = case_when(
    minsPerHour <= 45 ~ 0,
    minsPerHour > 45 ~ 1
  ))
songs_classification <- na.omit(songs_classification)

songs_classification <- songs_classification |>
  filter(!is.na(minsAbove45))
```

```
train_indexes <- as.vector(createDataPartition(songs_classification$minsAbove45, p = 0.8,
 ↪  list = FALSE))
music_train <- slice(songs_classification, train_indexes)
music_test <- slice(songs_classification, -train_indexes)

dim(songs_classification)
```

```
## [1] 1330    5
```

```
dim(music_train)
```

```
## [1] 1064    5
```

```
dim(music_test)
```

```
## [1] 266   5
```

```
fit <- glm(minsAbove45 ~ hour, data = music_train, family = "binomial")
```

```
threshold <- 0.2

predicted_prob_train <- predict(fit, newdata = music_train, type = "response")
predicted_prob_test <- predict(fit, newdata = music_test, type = "response")

predicted_class_train <- factor(if_else(predicted_prob_train > threshold, "1", "0"),
 ↪  levels = c("0", "1"))
predicted_class_test <- factor(if_else(predicted_prob_test > threshold, "1", "0"), levels
 ↪  = c("0", "1"))

music_train$minsAbove45 <- factor(music_train$minsAbove45, levels = c("0", "1"))
music_test$minsAbove45 <- factor(music_test$minsAbove45, levels = c("0", "1"))

confusion_matrix_train <- confusionMatrix(
  data = predicted_class_train,
  reference = music_train$minsAbove45,
  positive = "1"
)

confusion_matrix_test <- confusionMatrix(
  data = predicted_class_test,
```

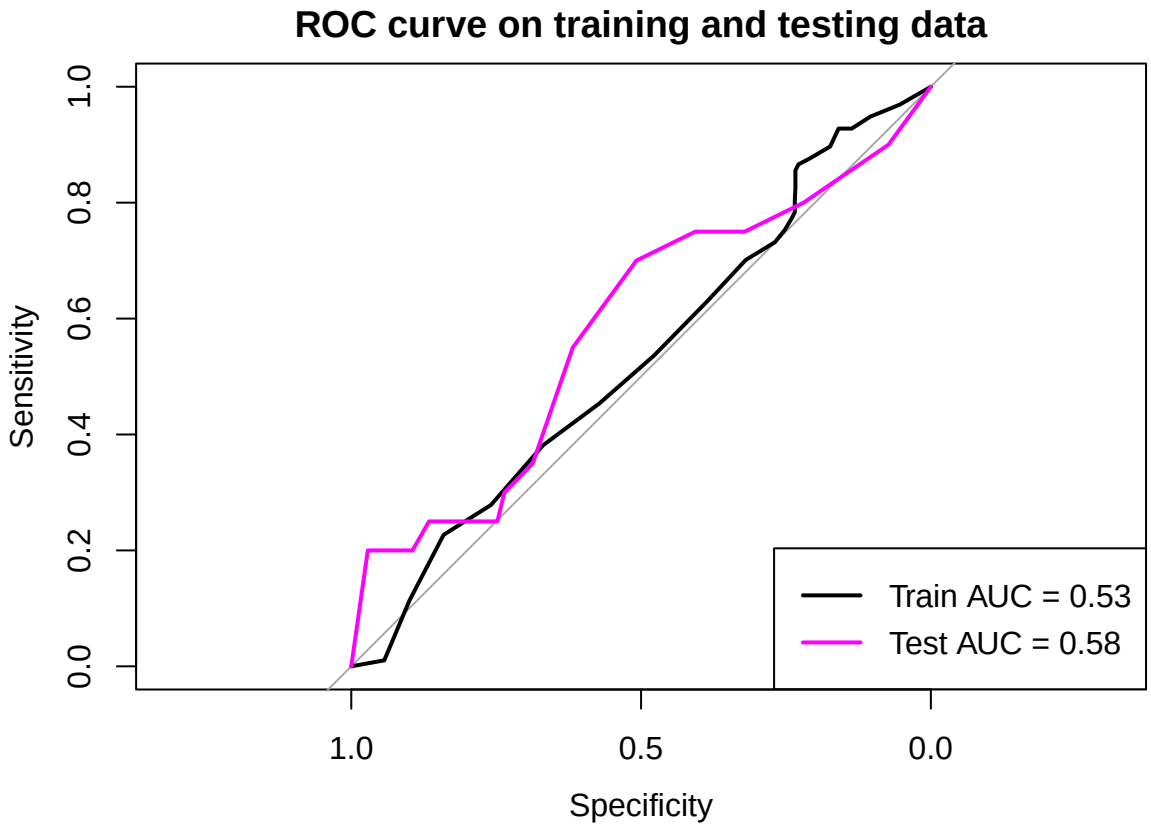
```
  reference = music_test$minsAbove45,
  positive = "1"
)

confusion_matrix_train$byClass["Sensitivity"]
```

```
## Sensitivity
##           0
```

```
confusion_matrix_test$byClass["Sensitivity"]
```

```
## Sensitivity
##           0
```

```
roc_train <- roc(music_train$minsAbove45, predicted_prob_train)
roc_test <- roc(music_test$minsAbove45, predicted_prob_test)

plot(roc_train, col = "black", lwd = 2, main = "ROC curve on training and testing data")
plot(roc_test, col = "magenta", lwd = 2, add = TRUE)

legend("bottomright",
 legend = c(
 paste("Train AUC =", round(auc(roc_train), 2)),
 paste("Test AUC =", round(auc(roc_test), 2))
),
 col = c("black", "magenta"),
 lwd = 2
)
```



The AUC for both our training and testing data is 0.51, which is really not any better than a completely random model. This let's us know that even though we used hour to get a decent model combined with month, on it's own, it's not a great predictor.

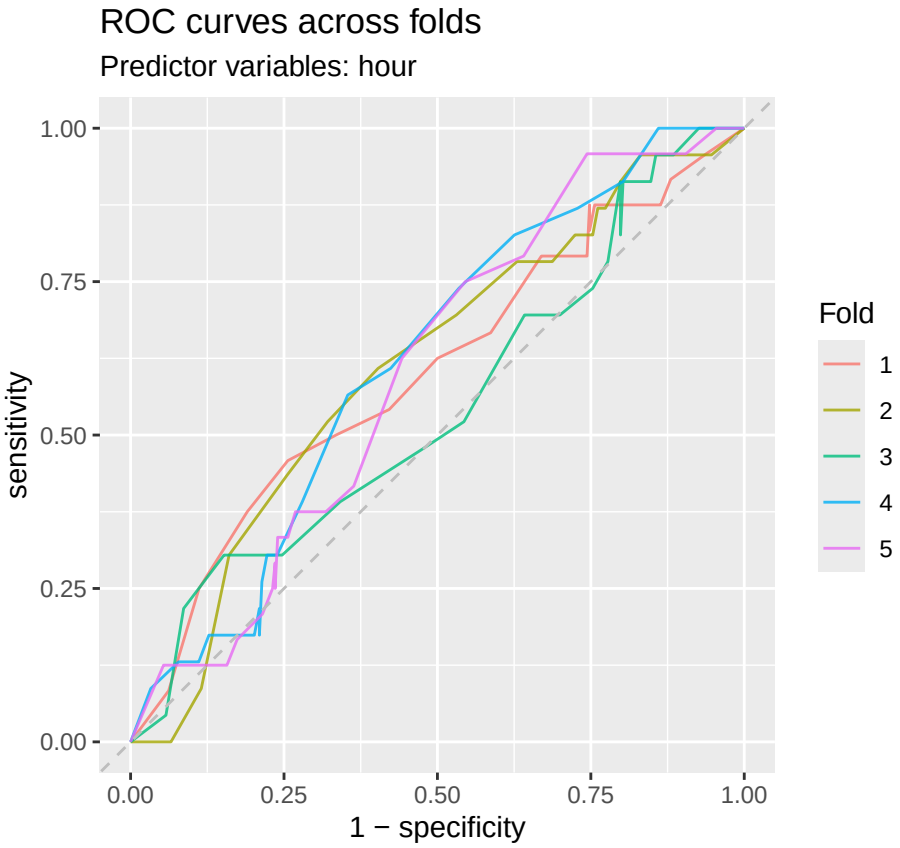
K-fold cross validation for just hour

```
songs_classification$minsAbove45 <- factor(
  songs_classification$minsAbove45,
  levels = c(0, 1),
  labels = c("no", "yes")
)

cv_control <- trainControl(
  method = "cv",
  number = 5,
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = "final"
)

cv_model <- train(
  minsAbove45 ~ hour,
  data = songs_classification,
  method = "glm",
  family = "binomial",
  trControl = cv_control
)

roc_list <- cv_model$pred |>
  group_by(Resample) |>
  group_map(~ roc(response = .x$obs, predictor = .x$yes, levels = c("no","yes")))
```

```
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
```

```
## Setting direction: controls > cases
## Setting direction: controls > cases
```

```
roc_df <- map_dfr(roc_list, ~ as_tibble(coords(.x, "all")), .id = "Fold")

ggplot(roc_df, aes(x = 1 - specificity, y = sensitivity, color = Fold)) +
  geom_line(alpha=0.8) +
 geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "gray") +
 labs(
   title = "ROC curves across folds",
   subtitle = "Predictor variables: hour") +
 coord_equal()
```



Our folds remain consistent, with some being under 0.5 (the dashed line), which is not ideal. However, it does show that our model remains consistent.

ROC curve with just month predictor

```
songs_classification <- songs_date |>
  mutate(minsAbove45 = case_when(
    minsPerHour <= 45 ~ 0,
    minsPerHour > 45 ~ 1
  ))
songs_classification <- na.omit(songs_classification)

songs_classification <- songs_classification |>
  filter(!is.na(minsAbove45))
```

```
train_indexes <- as.vector(createDataPartition(songs_classification$minsAbove45, p = 0.8,
 ↪  list = FALSE))
music_train <- slice(songs_classification, train_indexes)
music_test <- slice(songs_classification, -train_indexes)

dim(songs_classification)
```

```
## [1] 1330    5
```

```
dim(music_train)
```

```
## [1] 1064    5
```

```
dim(music_test)
```

```
## [1] 266   5
```

```
fit <- glm(minsAbove45 ~ month, data = music_train, family = "binomial")
```

```
threshold <- 0.2

predicted_prob_train <- predict(fit, newdata = music_train, type = "response")
predicted_prob_test <- predict(fit, newdata = music_test, type = "response")

predicted_class_train <- factor(if_else(predicted_prob_train > threshold, "1", "0"),
 ↪  levels = c("0", "1"))
predicted_class_test <- factor(if_else(predicted_prob_test > threshold, "1", "0"), levels
 ↪  = c("0", "1"))

music_train$minsAbove45 <- factor(music_train$minsAbove45, levels = c("0", "1"))
music_test$minsAbove45 <- factor(music_test$minsAbove45, levels = c("0", "1"))

confusion_matrix_train <- confusionMatrix(
  data = predicted_class_train,
  reference = music_train$minsAbove45,
  positive = "1"
)

confusion_matrix_test <- confusionMatrix(
  data = predicted_class_test,
  reference = music_test$minsAbove45,
  positive = "1"
)
```

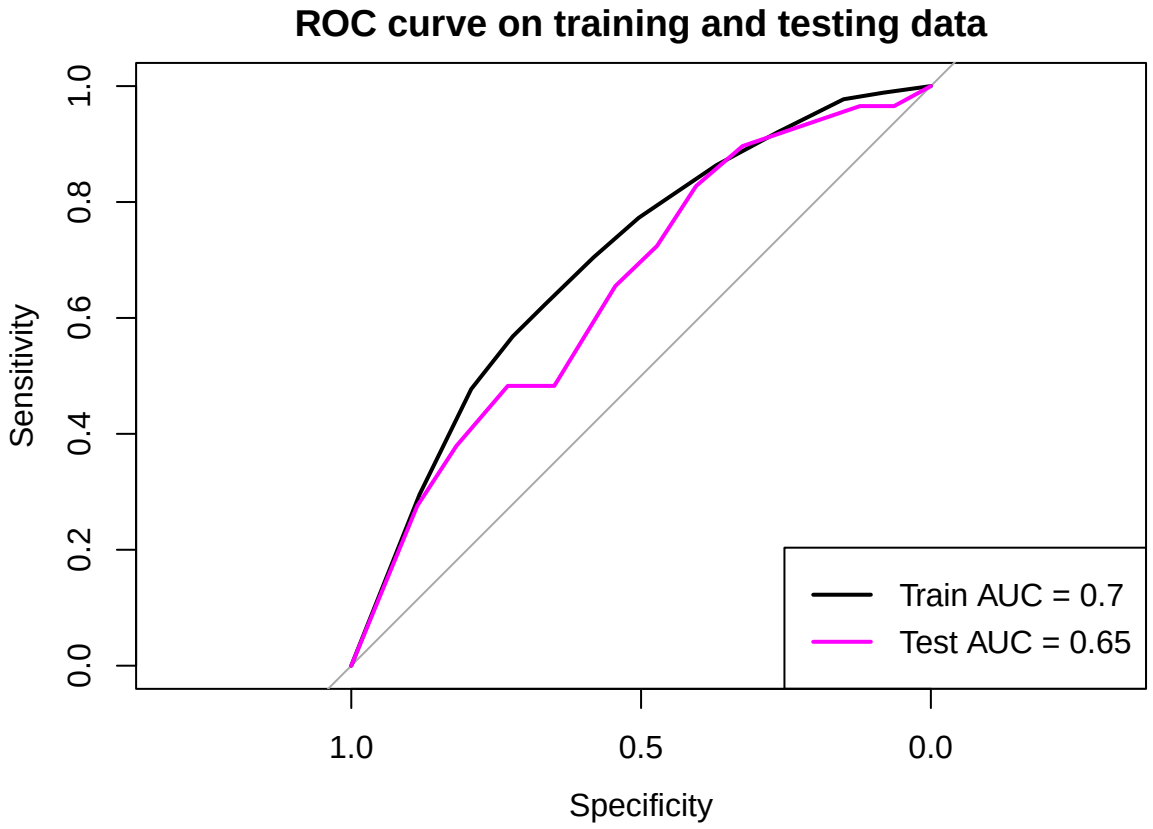
```r
confusion_matrix_train$byClass["Sensitivity"]
```

```
## Sensitivity
##           0
```

```r
confusion_matrix_test$byClass["Sensitivity"]
```

```
## Sensitivity
##           0
```

```r
roc_train <- roc(music_train$minsAbove45, predicted_prob_train)
roc_test <- roc(music_test$minsAbove45, predicted_prob_test)

plot(roc_train, col = "black", lwd = 2, main = "ROC curve on training and testing data")
plot(roc_test, col = "magenta", lwd = 2, add = TRUE)

legend("bottomright",
 legend = c(
 paste("Train AUC =", round(auc(roc_train), 2)),
 paste("Test AUC =", round(auc(roc_test), 2))
),
 col = c("black", "magenta"),
 lwd = 2
)
```



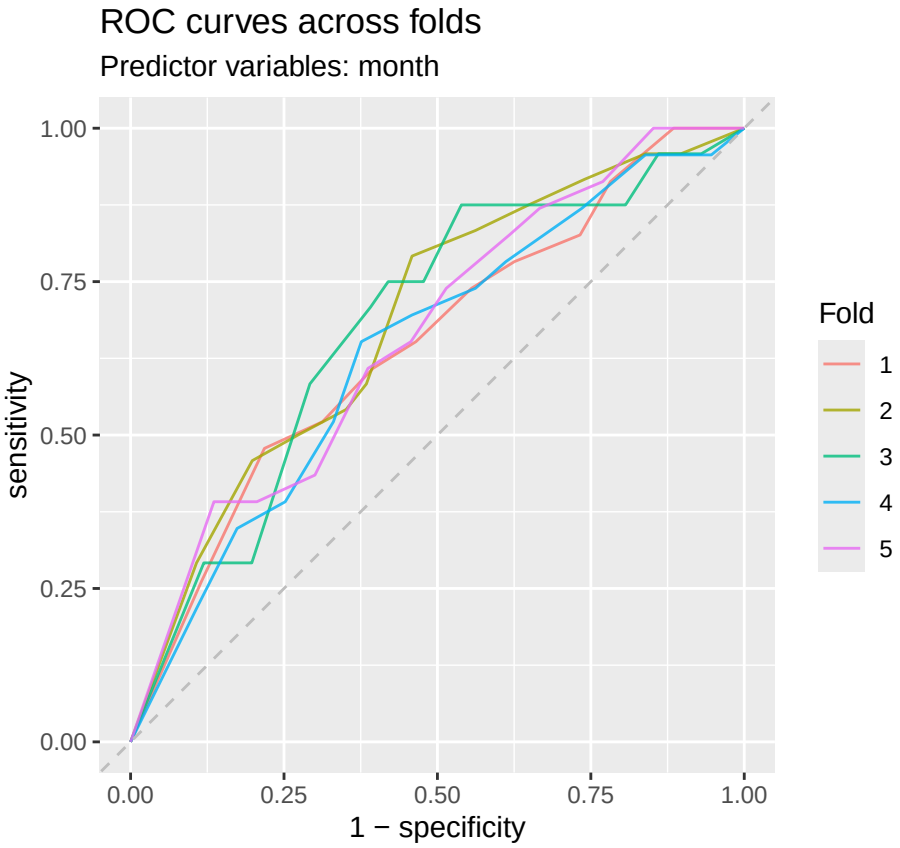
For just month, our model seems to be somewhat stronger than for month and hour combined, which makes sense considering what we learned about hour as a predictor.

K-Fold cross validation for just month

```
songs_classification$minsAbove45 <- factor(
  songs_classification$minsAbove45,
  levels = c(0, 1),
  labels = c("no", "yes")
)

cv_control <- trainControl(
  method = "cv",
  number = 5,
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = "final"
)

cv_model <- train(
  minsAbove45 ~ month,
  data = songs_classification,
  method = "glm",
  family = "binomial",
  trControl = cv_control
)

roc_list <- cv_model$pred |>
  group_by(Resample) |>
  group_map(~ roc(response = .x$obs, predictor = .x$yes, levels = c("no","yes")))
```

```
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
```

```
roc_df <- map_dfr(roc_list, ~ as_tibble(coords(.x, "all")), .id = "Fold")

ggplot(roc_df, aes(x = 1 - specificity, y = sensitivity, color = Fold)) +
  geom_line(alpha=0.8) +
 geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "gray") +
 labs(
   title = "ROC curves across folds",
   subtitle = "Predictor variables: month") +
 coord_equal()
```



Our ROC curves do seem to have some variation across the 5 folds, however, I don't think any of them has a significant drop in AUC. This model still performs the best.

Conclusion

Through this report, I was able to understand better my listening habits, and even create a model that could be used to future patterns. I found that there is a relatively significant impact of month of the year on the amount of time I listen to music, which makes sense because I tend to listen to music most when I'm doing schoolwork.

I would be interested in continuing this research further with more data on genres of music, and see if the type of music I listen to changes on hour and season.

I also find it very interesting that hour of day really didn't provide a strong model for predicting listening, even though we could visualize some of these trends in my heatmap.

I would also love to collect my own data next year and for years after, and continue doing this research so I could compare my listening habits from year to year, and see if the ways I listen to music change annually.

Ultimately, even though I was limited in the data that was available and accessible to me, I had a lot of fun considering patterns within myself and using the data I could to form meaningful patterns and connections.

Image sources

https://www.google.com/url?sa=i&url=https%3A%2F%2Fopen.spotify.com%2Fartist%2F4aKWmkWAKviFlyvHYPTNQY&psig=AOvVaw1PFF6X0TmZPyi9cXcq4Gay&ust=1746387411436000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCNCh75uGiI0DFQAAAAAdAAAAABAK

https://www.google.com/url?sa=i&url=https%3A%2F%2Fopen.spotify.com%2Fartist%2F1r1uxoy19fzMxunt3ONAkG&psig=AOvVaw2Uwp1uf4XbLNt7VWW6J-GR&ust=1746387509194000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCPCdwcqGiI0DFQAAAAAdAAAAABAE

https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.teamwass.com%2Fmusic%2Fadam-melchor%2F&psig=AOvVaw22O_Fsgm9XwwJtB5pkp3d6&ust=1746387536671000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCNjB5NeGiI0DFQAAAAAdAAAAABAE

https://www.google.com/url?sa=i&url=https%3A%2F%2Fopen.spotify.com%2Fartist%2F7GlBOeep6PqTfFi59PTUUN&psig=AOvVaw2BaoTAABAyeRYYjAWbaTew&ust=1746387561857000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCPiIh-SGiI0DFQAAAAAdAAAAABAE

https://www.google.com/url?sa=i&url=https%3A%2F%2Fvariety.com%2F2024%2Fdigital%2Fnews%2Ftaylor-swift-songs-return-tiktok-1235967738%2F&psig=AOvVaw13HZNjMB_vKlv6NNIA8b3t&ust=1746387628577000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCPiB04OHiI0DFQAAAAAdAAAAABAE

https://www.google.com/url?sa=i&url=https%3A%2F%2Fxboygeniusx.bandcamp.com%2Falbum%2Fthe-record&psig=AOvVaw3zDwABxPllt-Bc2QXW1dr_&ust=1746387678743000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCIjMtpuHiI0DFQAAAAAdAAAAABAE

https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.npr.org%2F2025%2F03%2F24%2Fnx-s1-5338777%2Flucy-dacus-forever-is-a-feeling-review-boygenius&psig=AOvVaw0EgqYxpxyoipjCdnggeS52&ust=1746387705981000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCNjr3aiHiI0DFQAAAAAdAAAAABAE

https://www.google.com/url?sa=i&url=https%3A%2F%2Ftophousetheband.com%2F&psig=AOvVaw1zEHVIIuijwW4bP0cSZg9y&ust=1746387737458000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCNDZqbeHiI0DFQAAAAAdAAAAABAE

https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.npr.org%2Fsections%2Fallsongs%2F2013%2F01%2F29%2F170552631%2Fwhy-ive-never-liked-fleetwood-macs-rumours&psig=AOvVaw3m0J2H5rGQsWAyHG_1dZTH&ust=1746387772178000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCJjF9ceHiI0DFQAAAAAdAAAAABAE

https://www.google.com/url?sa=i&url=https%3A%2F%2Fopen.spotify.com%2Fartist%2F2FXC3k01G6Gw61bmprjgqS&psig=AOvVaw3xpjVln4bN4iXbScd_7KWp&ust=1746387793758000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCKDEjtKHiI0DFQAAAAAdAAAAABAE